

南京大学讲座管理系统

项目报告

目录

1. 主要功能点和操作场景分析.....	3
1.1 主要功能点.....	3
1.2 操作场景分析.....	3
1.2.1 用户管理.....	3
1.2.2 学生功能.....	4
1.2.3 管理员功能.....	6
2. 两种架构选择及对应的架构图设计.....	7
2.1 MVC 架构.....	8
2.1.1 模块视图.....	8
2.1.3 组件-连接件视图.....	9
2.2 SOA 架构.....	10
2.2.1 模块视图.....	10
2.2.2 组件-连接件视图.....	11
3. 非功能需求及 ASR 描述.....	11
3.1 安全性.....	11
3.2 可用性.....	12
3.3 互操作性.....	12
3.4 可修改性.....	12
3.5 性能.....	13
3.6 可维护性.....	14
4. 系统类图设计.....	14
5. 组件/连接件到实现类的映射.....	15
6. 两种架构的比较及最终选择.....	15
6.1 整体比较.....	15
6.2 基于系统实际的最终选择.....	17
7. 具体实现技术的选择与解释.....	17
8. 基于 MVC 架构的 ADD 过程.....	18
8.1 第一次迭代结果.....	18
8.2 第二次迭代过程.....	19
8.2.1 识别所选模块的 ASR.....	19
8.2.2 每个 ASR 可选的设计决策.....	20
8.2.3 设计决策的选择及分析.....	20

8.2.4 第二次迭代结果.....	22
8.3 第三次迭代过程.....	22
8.3.1 识别所选模块的 ASR.....	22
8.3.2 每个 ASR 可选的设计决策.....	23
8.3.3 设计决策的选择及分析.....	24
8.3.4 第三次迭代结果.....	25
8.4 第四次迭代之后.....	25
9.基于 SOA 架构的 ADD 过程.....	26
9.1 第一次迭代结果.....	26
9.2 第二次迭代过程.....	26
9.2.1 识别所选模块的 ASR.....	27
9.2.2 每个 ASR 可选的设计决策.....	27
9.2.3 设计决策的选择及分析.....	28
9.2.4 第二次迭代结果.....	29
9.3 第三次迭代过程.....	29
9.3.1 识别所选模块的 ASR.....	29
9.3.2 每个 ASR 可选的设计决策.....	30
9.3.3 设计决策的选择及分析.....	31
9.3.4 第三次迭代结果.....	32
9.4 第四次迭代之后.....	32
10.ATAM 分析过程.....	32
10.1 质量属性效用树.....	32
10.2ATAM 分析.....	34
10.3 敏感点和权衡点.....	38
10.4 风险和非风险.....	39
11.挑战和经验.....	40
12.组员和分工.....	41

1.主要功能点和操作场景分析

1.1 主要功能点

- 用户管理
 - 注册
 - 登录
 - 用户权限管理
 - 个人信息修改
- 学生功能
 - 讲座推荐
 - 查看讲座信息
 - 讲座报名
 - 讲座退选
 - 讲座评价
 - 讨论讲座
 - 为收费讲座支付入场费
 - 收看允许录像的讲座的直播和回放
- 管理员功能
 - 导入、修改讲座信息
 - 强制报名退选讲座
 - 讲座反馈
 - 讨论区管理

1.2 操作场景分析

1.2.1 用户管理

表 1 登录/注册

场景要素	可能的值
刺激源	希望访问该系统的用户
刺激	用户希望访问该系统
制品	登录/注册子系统
环境	正常操作
响应	验证用户登录信息/注册用户
响应度量	响应时间不超过 0.5 秒。 用户的个人信息及密码应被有效保护，被攻击盗取的概率不大于 0.01%

表 2 用户权限管理

场景要素	可能的值
刺激源	管理员
刺激	管理员希望查看或修改用户权限
制品	用户权限管理子系统
环境	正常操作
响应	显示用户权限/保存用户权限修改设置
响应度量	响应时间不超过 0.5 秒。权限设置不应有误。

表 3 个人信息修改

场景要素	可能的值
刺激源	希望修改个人信息的用户
刺激	用户希望修改个人信息
制品	个人信息修改子系统
环境	正常操作
响应	修改个人信息并保存
响应度量	响应时间不超过 0.5 秒。个人信息存储不应出错。

1.2.2 学生功能

表 4 查看讲座信息

场景要素	可能的值
刺激源	学生用户
刺激	学生用户希望查看当前/个人的讲座信息
制品	查看讲座信息子系统
环境	正常操作
响应	显示当前/个人的讲座信息
响应度量	响应时间不超过 0.5 秒。讲座信息显示不应有误。

表 5 报名参加讲座

场景要素	可能的值
刺激源	学生用户
刺激	学生用户希望报名参加某讲座
制品	报名参加讲座子系统
环境	正常操作

响应	若名额未满，则学生用户报名成功；否则显示报名失败。
响应度量	系统应该允许 5000 个用户同时进行正常的访问、报名操作

表 6 讲座签到/支付

场景要素	可能的值
刺激源	学生用户
刺激	学生用户希望通过校园卡系统刷卡签到/支付入场费
制品	讲座签到/支付子系统
环境	正常操作
响应	报名参加某讲座的用户签到/支付成功
响应度量	响应时间不超过 0.5 秒。成功记录用户签到/支付信息的概率不低于 99.9%

表 7 讲座退选

场景要素	可能的值
刺激源	学生用户
刺激	学生用户希望退选已报名的某讲座
制品	讲座退选子系统
环境	正常操作
响应	若用户不是管理员强制设定必须参加某讲座，则用户退选讲座成功，并开放可报名参加讲座的名额；否则提示退选失败。
响应度量	响应时间不超过 0.5 秒。系统成功处理不同退选情况的概率不小于 99.99%

表 8 讲座讨论/评价

场景要素	可能的值
刺激源	学生用户
刺激	学生用户希望评价观看的某讲座并进行讨论
制品	讲座讨论/评价系统
环境	正常操作
响应	记录用户的讨论/评价信息
响应度量	响应时间不超过 0.5 秒。

表 9 讲座直播

场景要素	可能的值
刺激源	学生用户

刺激	学生用户希望观看某正在进行的讲座直播
制品	讲座直播子系统
环境	正常操作
响应	系统播放允许摄像的公开讲座的直播视频
响应度量	响应时间不超过 0.5 秒。 系统应该允许 500 个用户同时进行讲座直播视频观看。 视频系统直播延时不超过 0.5s

表 10 讲座回放

场景要素	可能的值
刺激源	学生用户
刺激	学生用户希望观看某已结束讲座的视频回放
制品	讲座回放子系统
环境	正常操作
响应	系统播放允许摄像的公开讲座的回放视频
响应度量	响应时间不超过 0.5 秒。 系统至少能够存储 500T 的视频数据

1.2.3 管理员功能

表 11 导入/修改讲座信息

场景要素	可能的值
刺激源	管理员用户
刺激	管理员希望导入/修改讲座信息
制品	讲座信息管理子系统
环境	正常操作
响应	系统保存导入/修改的讲座信息
响应度量	响应时间不超过 0.5 秒。 系统成功保存讲座信息的概率不小于 99.99%

表 12 强制报名/退选讲座

场景要素	可能的值
刺激源	管理员用户
刺激	管理员希望强制某些学生用户报名/退选讲座
制品	讲座报名管理子系统
环境	正常操作
响应	系统对应更改讲座及报名信息记录

响应度量	响应时间不超过 0.5 秒。 系统成功保存信息的概率不小于 99.99%
------	---

表 13 讲座反馈

场景要素	可能的值
刺激源	管理员用户
刺激	管理员希望查看讲座反馈信息
制品	讲座反馈子系统
环境	正常操作
响应	系统显示讲座反馈信息及学生参与情况
响应度量	响应时间不超过 0.5 秒。

表 14 讨论区管理

场景要素	可能的值
刺激源	管理员用户
刺激	管理员希望管理讨论区，进行删除/置顶讨论内容，并设置某些用户禁言
制品	讨论区管理子系统
环境	正常操作
响应	系统保存响应操作
响应度量	响应时间不超过 0.5 秒。

2.两种架构选择及对应的架构图设计

注：清晰的架构图设计详见附属的 images 文件夹。

2.1 MVC 架构

2.1.1 模块视图

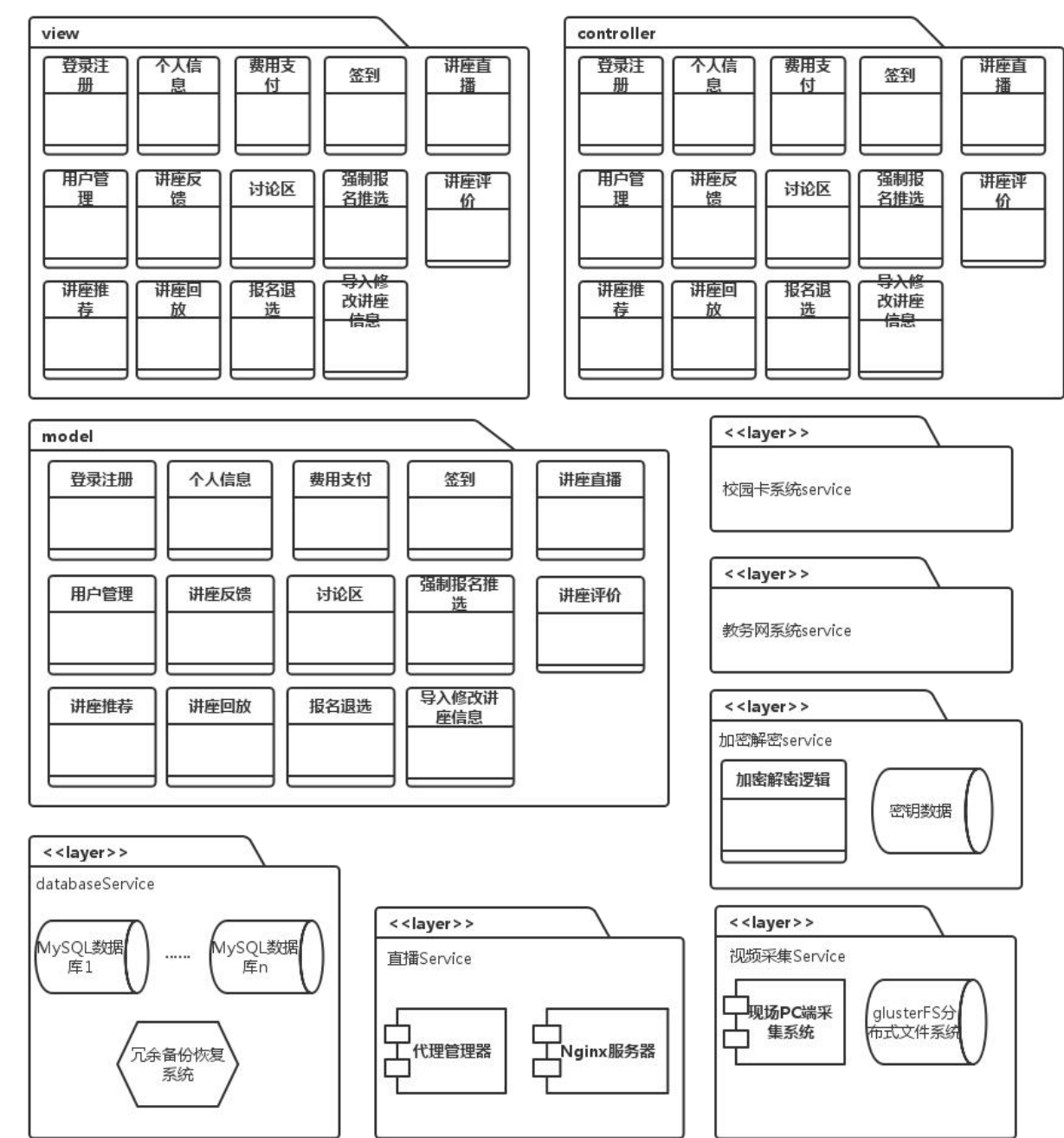


图 1： MVC 模块视图

2.1.3 组件-连接件视图

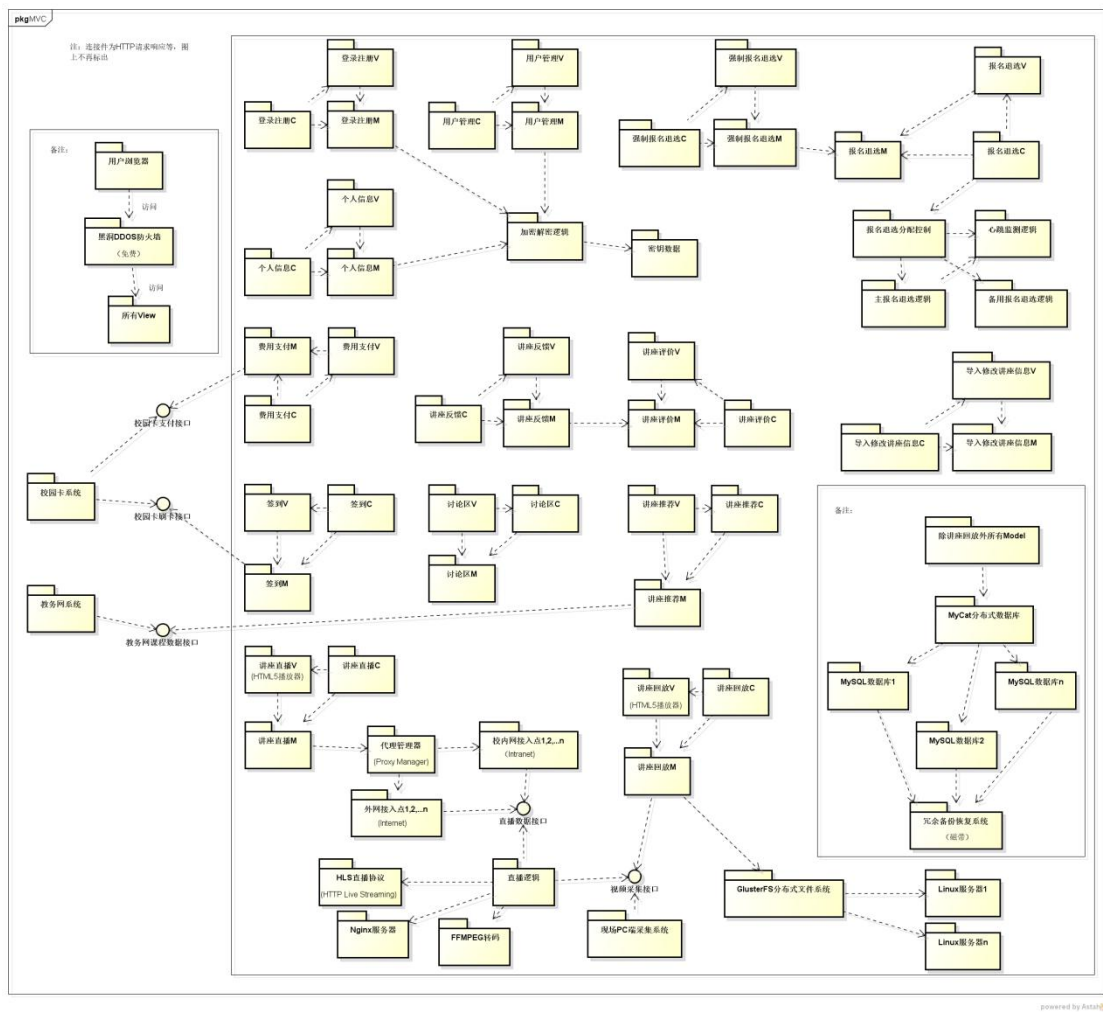


图 2: MVC 组件-连接器视图

2.2 SOA 架构

2.2.1 模块视图

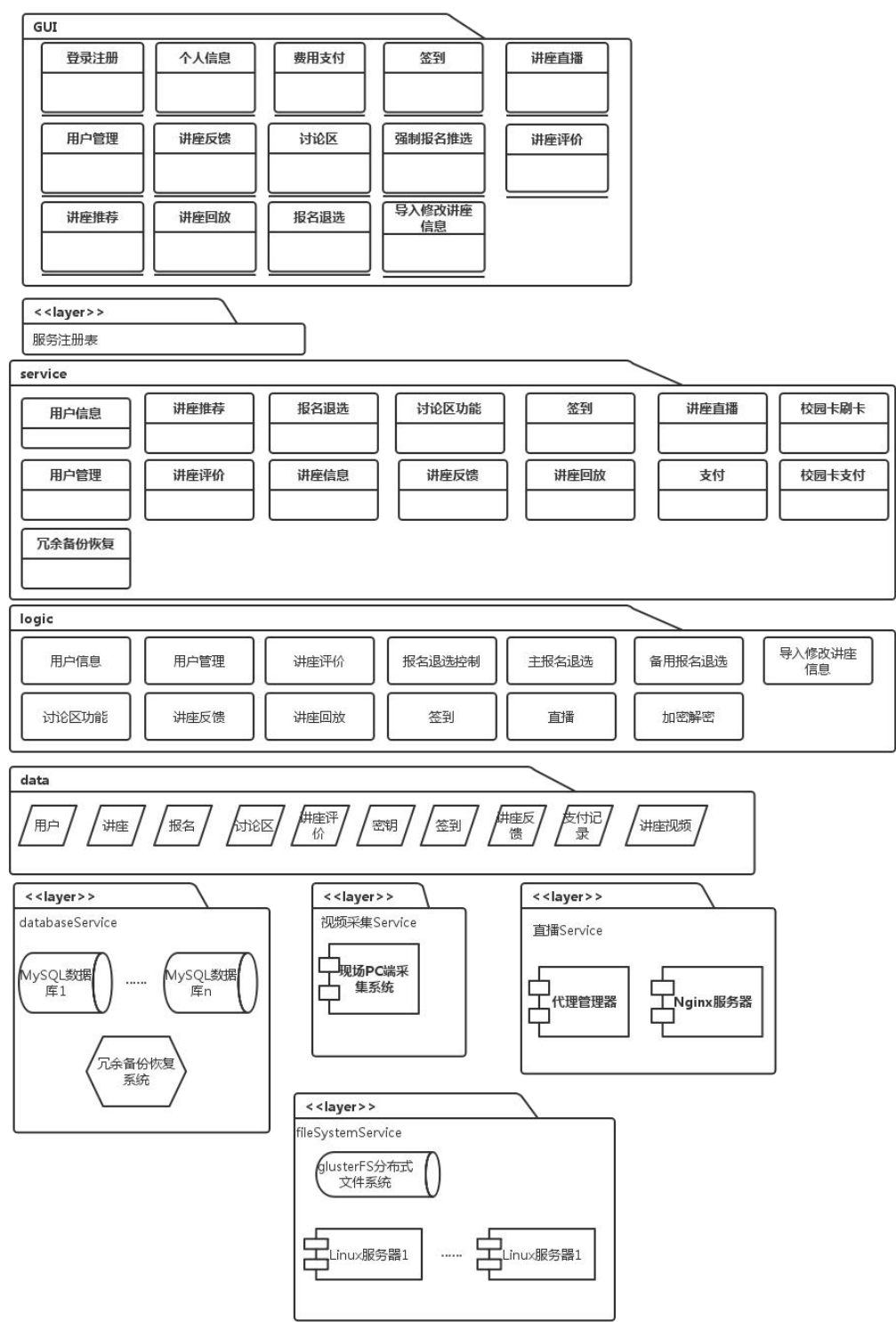


图 3：SOA 模块视图

2.2.2 组件-连接件视图

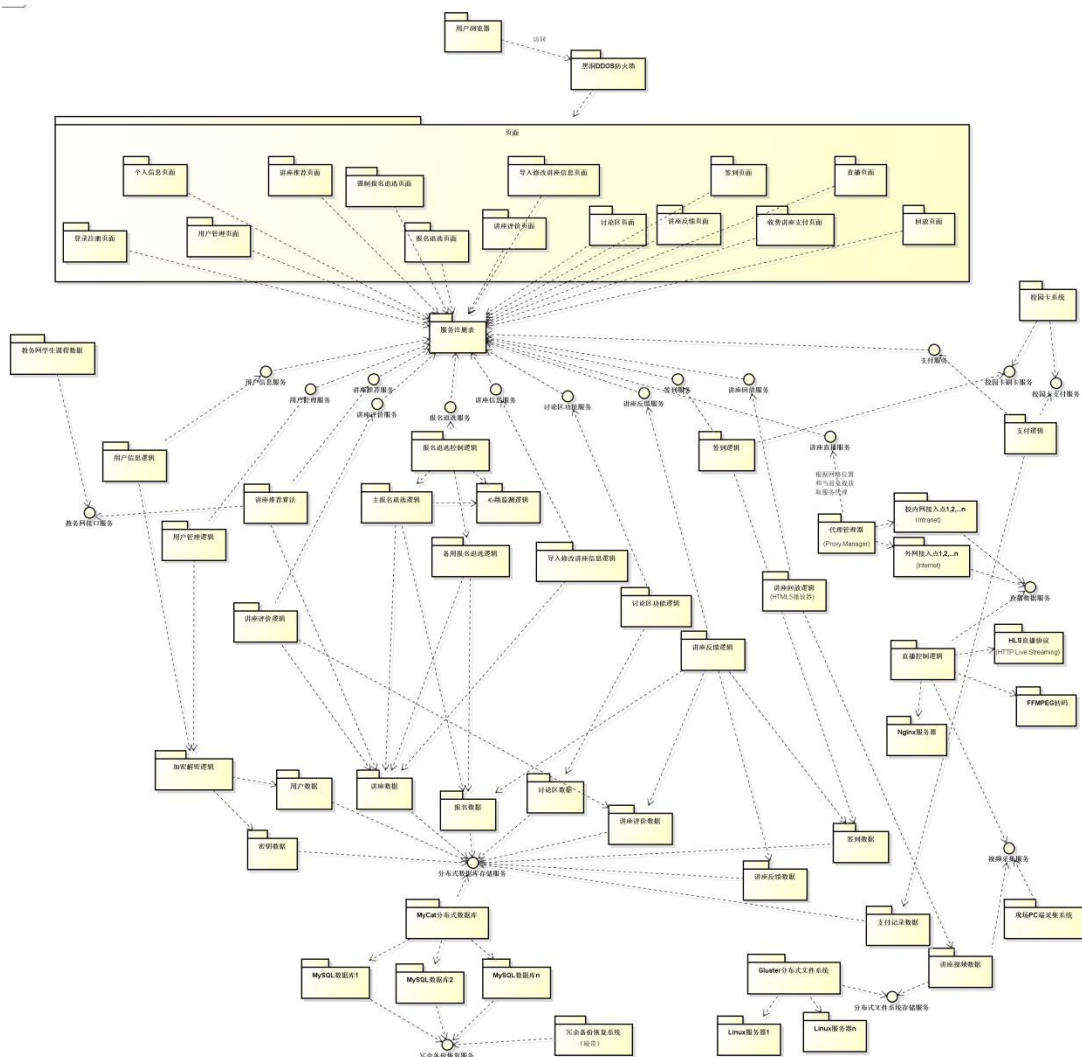


图 4：SOA 组件-连接器视图

3.非功能需求及 ASR 描述

3.1 安全性

Scenario 1

场景的部分	可能的值
源	来自内部/外部的经过了授权/未经过授权的个人或系统
刺激	试图修改/删除数据，访问系统服务，窃取用户信息
制品	系统服务、系统中的数据
环境	在线/离线，联网/断网，连接有防火墙或直接连接到了网络上

响应	对用户进行验证；加密用户的账户信息；阻止未授权用户访问；
响应度量	未认证用户无法访问数据和发布控制指令 数据被恶意修改/删除后可以在 10min 内进行恢复

3.2 可用性

Scenario 2

场景的部分	可能的值
源	来自授权的个人用户
刺激	用户希望使用系统功能，尤其是讲座报名、直播等
制品	整个系统
环境	系统运行时，尤其是高峰期
响应	系统能够为用户提供服务而不能频繁崩溃
响应度量	系统崩溃频率不超过 1 次/周，并能在崩溃发生后 1 分钟之内记录错误日志并恢复正常功能。

3.3 互操作性

Scenario 3

场景的部分	可能的值
源	系统发起/收到与校园卡系统/教务网系统/现场 PC 端采集系统的请求/响应
刺激	希望与上述的系统进行请求/响应 数据交换
制品	希望进行互操作的上述系统
环境	希望进行互操作的上述系统正在运行
响应	请求被正确获取并成功交换数据
响应度量	交换信息数据成功的概率不低于 99%

3.4 可修改性

Scenario 4

场景的部分	可能的值
源	开发者
刺激	开发者希望修改系统用户界面、数据标准、控制逻辑等
制品	代码

环境	设计、开发、维护系统时
响应	需要修改的模块被正确的修改，并不影响其他功能的实现
响应度量	每个模块的修改可以在 2 人月内完成 修改预算不超过总预算的 10% 不影响无关的系统功能

3.5 性能

性能场景 1：负载

Scenario 5

场景的部分	可能的值
源	访问系统的学生个人用户
刺激	学生用户希望在系统上进行讲座报名操作/查看讲座直播
制品	系统报名参加讲座子系统/讲座直播子系统
环境	联网状态
响应	系统能够正常工作，为每一个用户提供响应
响应度量	系统应该允许 5000 个用户同时进行正常的访问、报名操作/观看讲座直播 用户访问所需页面的时间不超过 0.1s

性能场景 2：容量

Scenario 6

场景的部分	可能的值
源	系统
刺激	系统希望进行讲座视频的存储
制品	存储视频数据的系统数据库
环境	运行时
响应	系统正确、完整、及时的存储大量讲座视频数据
响应度量	系统至少能够存储 50T 的视频数据，保持数据的完整性、正确性、一致性

性能场景 3：实时性

Scenario 7

场景的部分	可能的值
源	访问系统的学生个人用户
刺激	学生用户希望在系统上观看讲座直播
制品	系统讲座直播子系统
环境	联网状态

响应	系统直播视频清晰流畅，延时短
响应度量	系统直播延时不超过 5s

3.6 可维护性

Scenario 8

场景的部分	可能的值
源	系统开发者
刺激	系统开发者希望对系统进行维护，修改其部分功能
制品	系统各个功能模块
环境	系统维护时
响应	及时、方便的进行修改，并对无关模块不造成影响
响应度量	每个模块的修改时间不超过 5 人日

4.系统类图设计

注：清晰的系统类图设计详见附属的 images 文件夹。

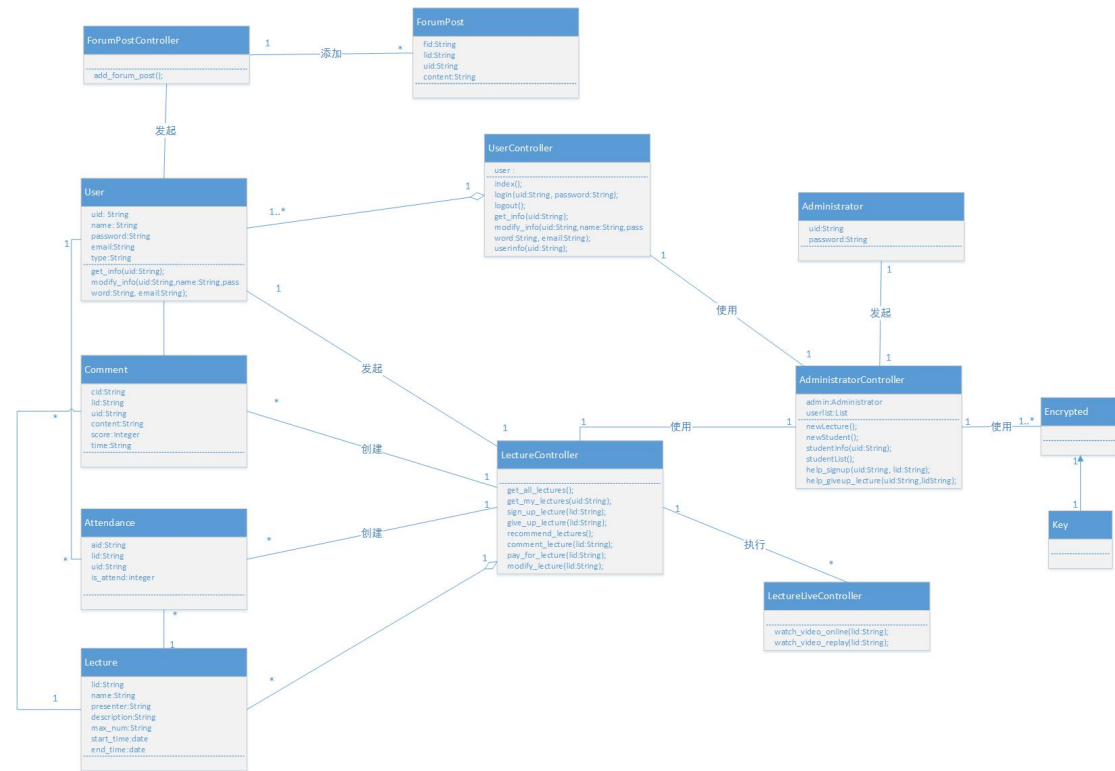


图 5：系统类图

5.组件/连接件到实现类的映射

实现类	组件连接器
UserController	登陆注册 C 个人信息 C
User	个人信息 M
Administrator	
AdministratorController	用户管理 C 强制报名退选 C 导入修改讲座信息 C
Lecture	讲座直播 M 讲座回放 M
LectureController	报名退选 C 费用支付 C 讲座反馈 C 讲座评价 C 讲座推荐 C 讲座出席 C
Encrypted	加密解密
Key	密钥数据
Comment	讲座评价 M
Attendance	讲座出席 M
LectureLiveController	讲座直播 C 讲座回放 C
ForumPost	讨论区 M
ForumPostController	讨论区 C

6.两种架构的比较及最终选择

6.1 整体比较

比较方面	SOA		MVC	
	pros	cons	pros	cons
安全性		SOA 架构的松耦合性和开放性带来安全性的问题。本系统涉及到安全性攸关的校园卡服务，使	MVC 没有安全性方面的明显缺点	

		用 SOA 会增加安全方案的复杂度。		
可用性	两种架构在可用性上没有大的区别			
互操作性	本系统涉及到即将建设的系统和已存在的校园卡系统和教务系统，可能存在平台和异构问题。选择接口而不是语言具体的类以及基于消息交互，极大的提高了系统的互操作性。			MVC 在互操作性上没有特殊的优势
可修改性	SOA 不同服务之间保持了一种无依赖的低耦合关系；服务本身是通过统一的接口定义语言来描述具体的服务内容，并且很好地封装了底层的具体实现。所以，当发生修改时，服务之间不会影响，并且底层实现不会影响调用。		MVC 架构耦合性低，MVC 的三个部件相对独立，尤其适用于 View 变更频繁的 Web 系统。	
性能：负载	两种架构没有特别大的差别。			
性能：容量	两种架构没有特别大的差别。			
性能：实时性		SOA 架构的性能稍低，主要是因为 SOA 的分布性质和 web 服务协议开销。SOA 的消息传递往往需要层层递进，如服务消费者-ESB-提供者，造成性能损失。	M、V、C 之间消息传递灵活，优于 SOA。	

可维护性	服务带来低耦合性和很好的模块化。		分离 M、V、C 也使得 WEB 应用更易于维护和修改。	
可扩展性	SOA 的可扩展性很好，容易构建应用服务体系			MVC 的可扩展性较弱
成本约束		开发工作较为繁杂，成本高	容易掌握，开发快速	

6.2 基于系统实际的最终选择

系统最终选择 MVC 架构，这一决策是考虑项目前景和约束而做出的。

本系统是学校使用的、单独的 Web 系统，本身不是某个应用服务体系的一部分，并不是像“某某投资银行的企业应用体系中的下单系统”这样的应用，未来也没有为其开发配套系统的预期，考虑这种情况，SOA 最突出的可扩展性的意义不大，而 SOA 的安全策略复杂、消息传递效率低、开发周期长、成本高的缺点却是显著的。与之相对的，MVC 的简单、快速、高效成为其最大的优势。

一言以蔽之，考虑项目实际，SOA 的优点体现不出来，其缺点却充分暴露。权衡之下，我们最终选择的是 MVC 架构。

7.具体实现技术的选择与解释

类别	实现方式	选择与解释
安全保障	用户数据加密	采用。成熟的安全手段，避免泄露用户隐私
	自动攻击检测	未采用。系统并不是网银系统等安全攸关系统，无需以性能为代价寻求过度的安全保护。
	检测到可疑访问后拒绝服务	未采用。可能因为误判而影响正常用户对系统的访问，降低系统可用性
可用性措施	DDOS 防火墙	采用。校园环境下 PC 密度大且安全意识淡薄，攻击者容易劫持大量肉鸡对系统进行拒绝服务攻击。
	选课模块进行备份和心跳监测	采用。系统需要应对热门讲座报名的高峰期，此时服务不容中断。心跳机制性能负担较小。
提高负载措施	直播双线接入	采用。可以充分利用高速的校园内网资源。
	增加计算资源	未采用。会增加成本。
提高容量措施	MyCat+MySQL 分布式数据库	采用。开源，虽然没有商家提供成套解决方案，但是没有直接成本。虽然需要开发者和维护者付出较多时间精力，但是对于学校应用的项目而言并不缺乏有技术能力的学生和教师，而且学生和教师的时间和精力比较廉价。

	Oracle 数据库解决方案	未采用。成本高。虽然能够节省时间精力，但是也失去了锻炼学生能力的机会。
	Linux + Gluster 分布式文件系统	采用。开源，没有直接成本，应用广泛。
	Windows Server+ DFS 文件系统	未采用。虽然能够节省时间精力，但是授权费用高，不能接受。
直播协议	HLS 协议	采用。虽然直播延迟稍高，但是兼容性好，能够兼容 iPhone。
	RTMP 协议	未采用。虽然直播延迟低，但是不兼容 iPhone，这在 iPhone 用户多的大学环境下是不可接受的。
视频转码	FFMPEG 转码	采用。性能较高，兼容性好。同时它包含了非常先进的音频/视频编解码库 libavcodec，保证了高可移植性。
直播服务器	Nginx 服务器	采用。轻量级服务器，技术成熟，开源，性能高。
数据备份	磁带机	采用。成本低，技术成熟，操作简单，容量大。
	双机热备	未采用。成本高，超出了系统对数据安全性的要求。

8.基于 MVC 架构的 ADD 过程

8.1 第一次迭代结果

采用 MVC 的架构，参考所需的功能需求，形成第一次迭代后的整体设计图。由于该系统的功能性需求整体较为清晰，因此在第一次迭代结束后，已经将大部分功能性需求纳入架构设计考虑。

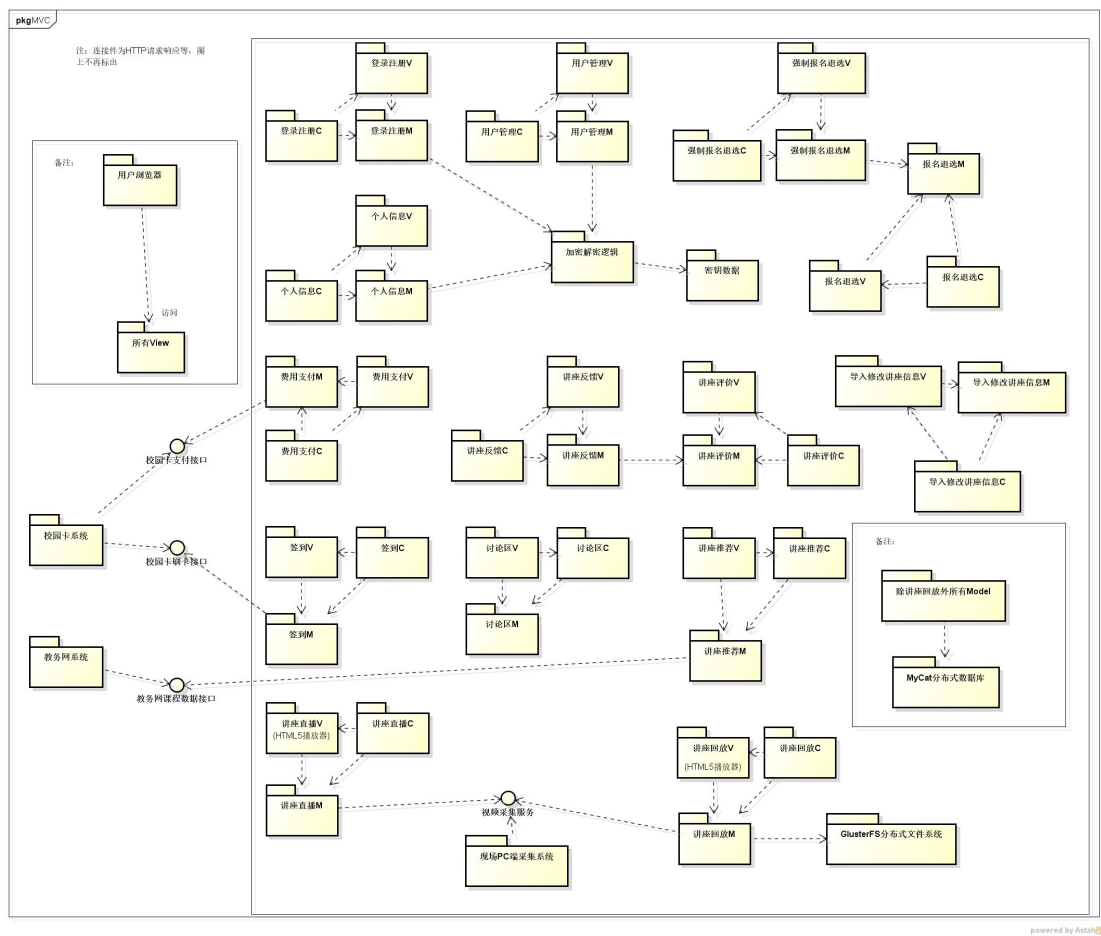


图 6：MVC 第一次迭代结果

8.2 第二次迭代过程

选取直播模块作为系统元素进行分解。在质量属性方面，该模块与互操作性、可修改性、负载性能、实时性有密切关系。

8.2.1 识别所选模块的 ASR

#	Architectural Drivers	Importance	Difficulty
1	Scenario 3: 互操作性: 系统发起现场 PC 端采集系统的请求被正确获取并成功交换数据	high	medium
2	Scenario 4: 可修改性	medium	low
3	Scenario 5: 负载性能: 系统能够允许 500 个用户同时观看讲座直播	high	high
4	Scenario 7: 实时性	high	high

8.2.2 每个 ASR 可选的设计决策

➤ 互操作性

可选的设计决策或模式	分析
在提供的服务列表中定位所需要的服务	主要针对所需的服务较多且有层次性的情况
使用控制机制来管理、协调服务的调用	针对所要服务的访问需求量较多的情况
抽取并剪裁接口	针对请求响应交换的数据不一致情况

➤ 可修改性

可选的设计决策或模式	分析
划分模块	可以有效缩小要修改的范围
抽象通用服务	将多个模块都需要的服务提取出来，形成可供多个模块共同使用的通用服务
添加接口	将功能和实现分离，隔离需要更改的模块
运行时注册	支持即插即用操作，但需要管理注册的额外开销

➤ 负载性能

可选的设计决策或模式	分析
维持计算的多个副本	需要考虑如何使副本保持一致和同步
增加可用资源	使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本
区分校内校外网络接入	可以充分利用高速廉价的内网资源，但是增加了系统的复杂性

➤ 实时性

可选的设计决策或模式	分析
使用 RTMP (Real Time Messaging Protocol) 协议	适合长时间播放，延时较低，但是又累积延时；是 Adobe 开发的协议，无法再 iPhone 中兼容
使用 HLS (HTTP Live Streaming) 协议	兼容性比 RTMP 好，但是延时比其高

8.2.3 设计决策的选择及分析

➤ 互操作性

可选的设计决策或模式	决策理由
------------	------

在提供的服务列表中定位所需要的服务	不采用。仅是关于直播的服务。
使用控制机制来管理、协调服务的调用	不采用。仅是关于直播的服务。
抽取并剪裁接口	采用。用于统一不同的数据格式。

➤ 可修改性

可选的设计决策或模式	决策理由
划分模块	采用。是一种常用且有效的策略。
抽象通用服务	不采用。并没有通用服务需要被抽象。
添加接口	采用。增加可修改性常用的策略。
运行时注册	不采用。会带来额外的管理开销。

➤ 负载性能

可选的设计决策或模式	决策理由
维持计算的多个副本	不采用。考虑同步需要的代价太大。
增加可用资源	不采用。会增加较多成本。
区分校内校外网络接入	采用。校园内网速率远远优于因特网，需要充分利用其资源。

➤ 实时性

可选的设计决策或模式	决策理由
使用 RTMP (Real Time Messaging Protocol) 协议	不采用。并没有很好的兼容性。大学生使用 iPhone 较多，RTMP 不支持 iPhone 的特性是不可接受的。
使用 HLS (HTTP Live Streaming) 协议	采用。该技术成熟且容易使用。此外，开源有益于控制开发系统的成本。

8.2.4 第二次迭代结果

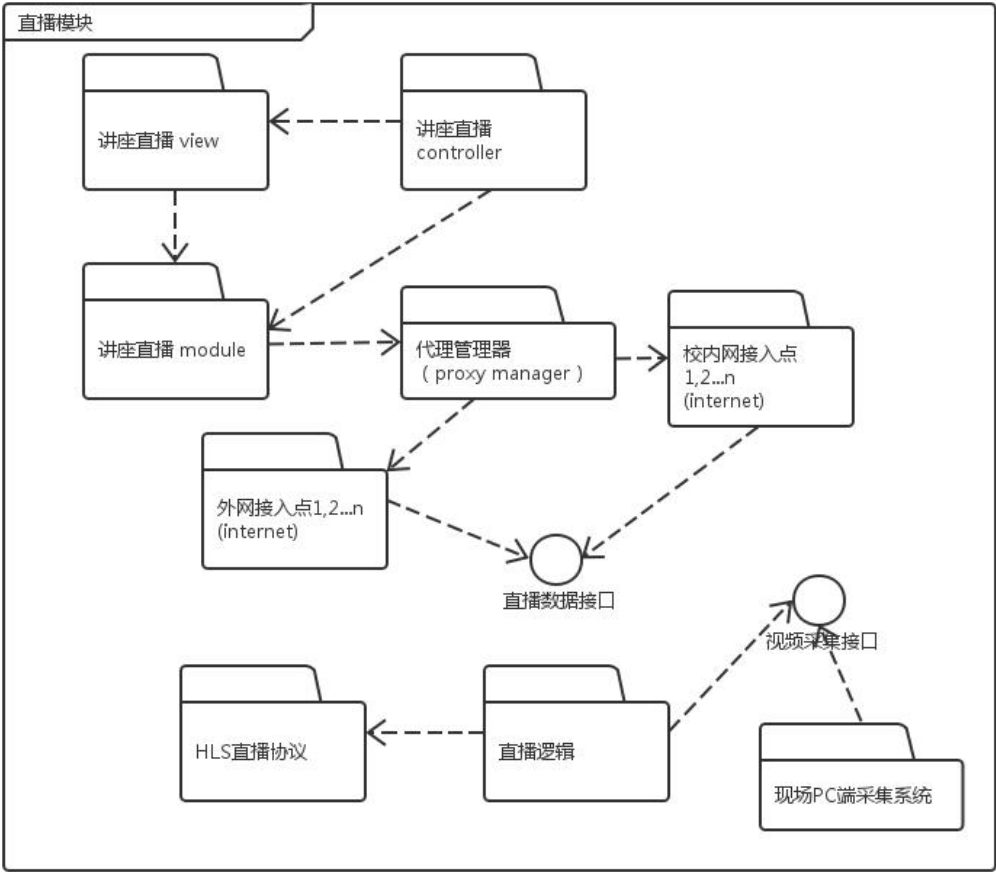


图 7：MVC 第二次迭代结果

8.3 第三次迭代过程

选取讲座报名模块作为系统元素进行分解。在质量属性方面，该模块与可获得性、可修改性、负载性能有密切关系。

8.3.1 识别所选模块的 ASR

#	Architectural Drivers	Importance	Difficulty
1	Scenario 2: 可获得性	high	medium
2	Scenario 4: 可修改性	medium	low
3	Scenario 5: 负载性能: 系统能够允许 500 个用户同时进行正	high	high

	常的访问、报名操作		
--	-----------	--	--

8.3.2 每个 ASR 可选的设计决策

➤ 可获得性

可选的设计决策或模式	分析
命令/响应	一个组件发出一个命令，并希望在预定义的时间内收到一个来自审查组件的响应。与对所有进程发出命令的远程错误探测器相比，这种策略所使用的通信带宽更少
心跳	一个组件定期发出一个心跳信息，另一个组件收听该信息。可以在起到心跳的同时传递数据。
主动冗余（热启动）	所有的冗余组件都以并行的方式对事件作出响应。错误发生时，使用该战术的系统的停机时间通常是几毫秒，因为备份是最新的，所以恢复所需的时间就是切换时间。
被动冗余	一个组件（主要的）对事件作出响应，并通知其他组件（备用的）必须进行的状态更新。该战术依赖于能够可靠地接管工作的备用组件。停机时间通常为几秒钟。
备件	备用件是计算平台配置用于更换各种不同的故障组件。当出现故障时，必须将其重新启动为适当的软件配置，并对其状态进行初始化。该战术的停机时间通常为几分钟。

➤ 可修改性

可选的设计决策或模式	分析
划分模块	可以有效缩小要修改的范围
抽象通用服务	将多个模块都需要的服务提取出来，形成可供多个模块共同使用的通用服务
添加接口	将功能和实现分离，隔离需要更改的模块
运行时注册	支持即插即用操作，但需要管理注册的额外开销

➤ 负载性能

可选的设计决策或模式	分析
维持计算的多个副本	需要考虑如何使副本保持一致和同步
增加可用资源	使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本

8.3.3 设计决策的选择及分析

➤ 可获得性

可选的设计决策或模式	决策理由
命令/响应	不采用。
心跳	采用。常用的方式，且使系统的可获得性更高。
主动冗余（热启动）	采用。该系统对于选课报名的可获得性要求很高，采用热备份可以获得更高的可获得性。
被动冗余	不采用。停机时间过长，对于讲座报名这种短时间内对可获得性要求极高的子系统而言并不够好。
备件	不采用。理由同上。

➤ 可修改性

可选的设计决策或模式	决策理由
划分模块	采用。是一种常用且有效的策略。
抽象通用服务	不采用。并没有通用服务需要被抽象。
添加接口	不采用。不适用于该子系统。
运行时注册	不采用。会带来额外的管理开销。

➤ 负载性能

可选的设计决策或模式	决策理由
维持计算的多个副本	采用。保证在一个计算逻辑失效后可以在较短的时间内启用备用逻辑。
增加可用资源	不采用。会增加较多成本。

8.3.4 第三次迭代结果

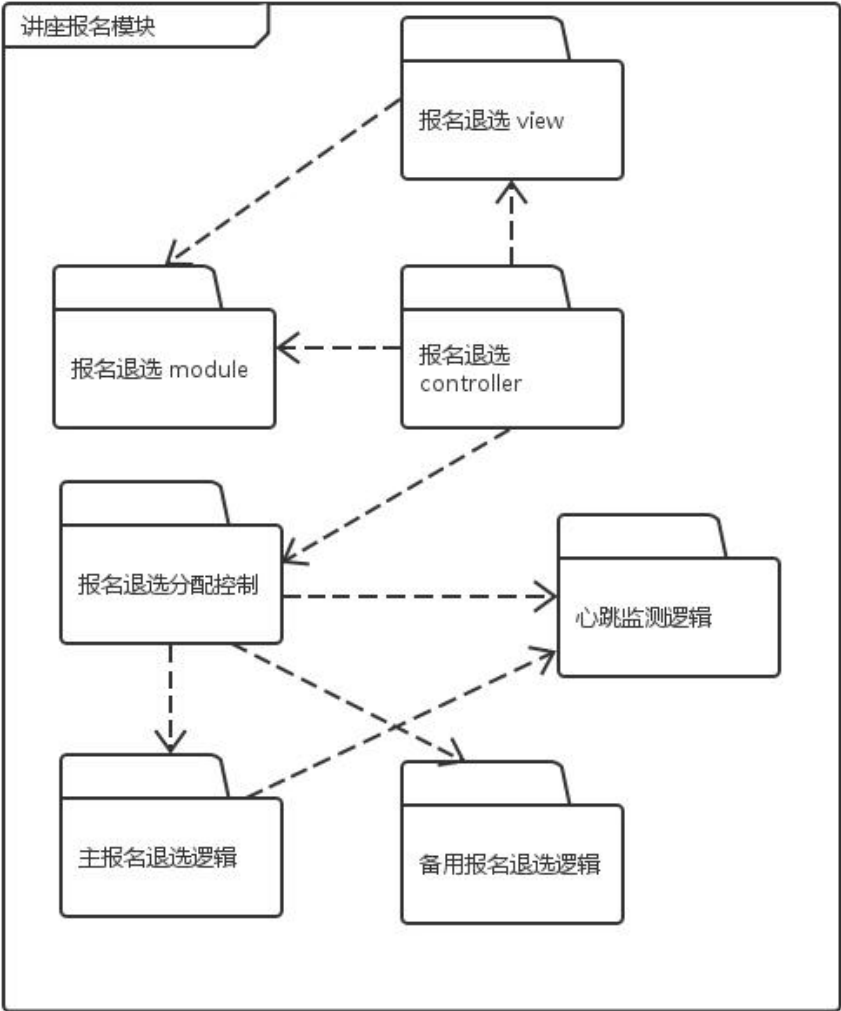


图 8：MVC 第三次迭代结果

8.4 第四次迭代之后

受篇幅所限，第四次迭代直至完成架构的过程此处不再给出，最终结果请参照第 2 节。

9.基于 SOA 架构的 ADD 过程

9.1 第一次迭代结果

采用 SOA 的架构，参考所需的功能需求，形成第一次迭代后的整体设计图。由于该系统的功能性需求整体较为清晰，因此在第一次迭代结束后，已经将大部分功能性需求纳入架构设计考虑。

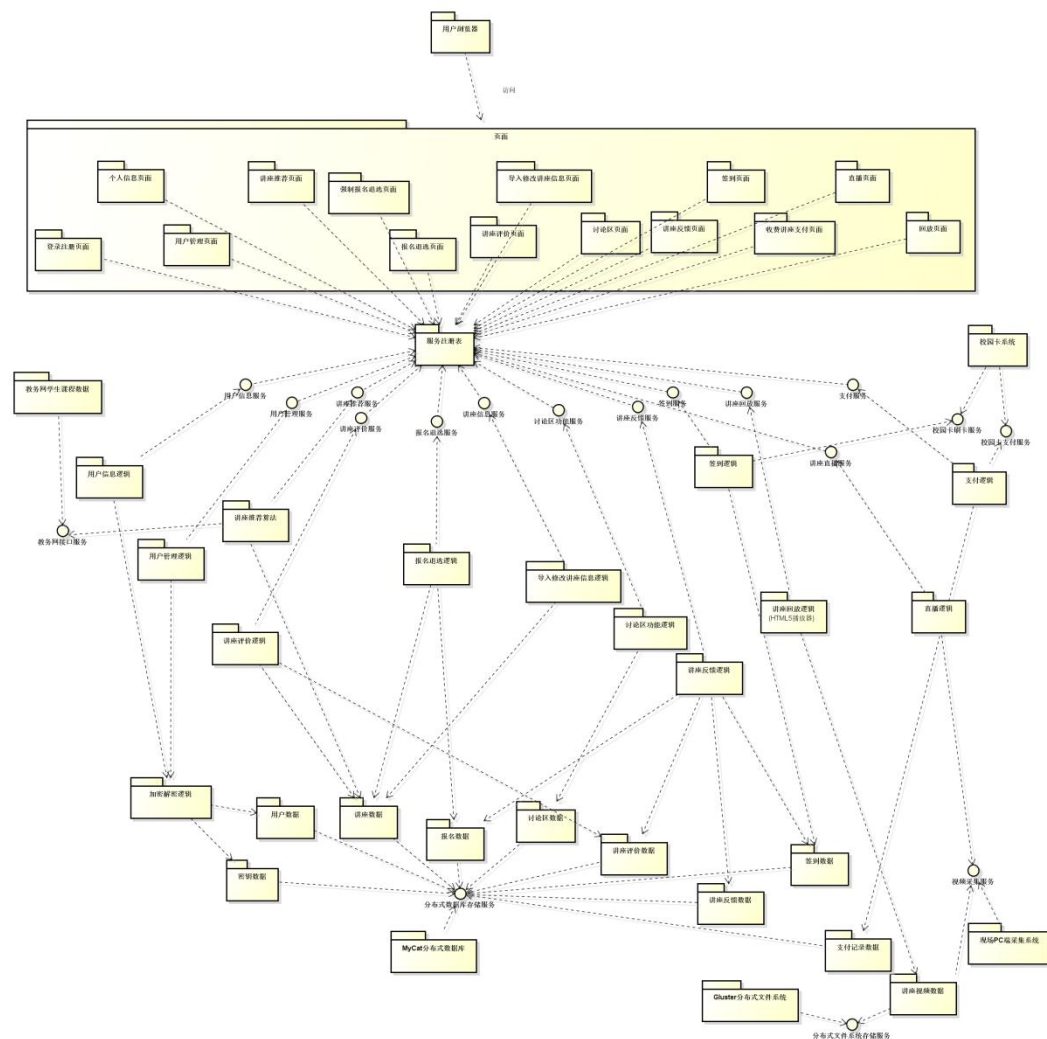


图 9: SOA 第一次迭代结果

9.2 第二次迭代过程

选取直播模块作为系统元素进行分解。在质量属性方面,该模块与互操作性、可修改性、负载性能、实时性有密切关系。

9.2.1 识别所选模块的 ASR

#	Architectural Drivers	Importance	Difficulty
1	Scenario 3: 互操作性: 系统发起现场 PC 端采集系统的请求被正确获取并成功交换数据	high	medium
2	Scenario 4: 可修改性	medium	low
3	Scenario 5: 负载性能: 系统能够允许 500 个用户同时观看讲座直播	high	high
4	Scenario 7: 实时性	high	high

9.2.2 每个 ASR 可选的设计决策

➤ 互操作性

可选的设计决策或模式	分析
在提供的服务列表中定位所需要的服务	主要针对所需的服务较多且有层次性的情况
使用控制机制来管理、协调服务的调用	针对所要服务的访问需求量较多的情况
抽取并剪裁接口	针对请求响应交换的数据不一致情况

➤ 可修改性

可选的设计决策或模式	分析
划分模块	可以有效缩小要修改的范围
抽象通用服务	将多个模块都需要的服务提取出来, 形成可供多个模块共同使用的通用服务
添加接口	将功能和实现分离, 隔离需要更改的模块
运行时注册	支持即插即用操作, 但需要管理注册的额外开销

➤ 负载性能

可选的设计决策或模式	分析
维持计算的多个副本	需要考虑如何使副本保持一致和同步
增加可用资源	使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本
区分校内校外网络接入	可以充分利用高速廉价的内网资源, 但是增加了系统的复杂性

➤ 实时性

可选的设计决策或模式	分析
使用 RTMP (Real Time Messaging Protocol) 协议	适合长时间播放，延时较低，但是又累积延时；是 Adobe 开发的协议，无法再 iPhone 中兼容
使用 HLS (HTTP Live Streaming) 协议	兼容性比 RTMP 好，但是延时比其高

9.2.3 设计决策的选择及分析

➤ 互操作性

可选的设计决策或模式	决策理由
在提供的服务列表中定位所需要的服务	不采用。仅是关于直播的服务。
使用控制机制来管理、协调服务的调用	不采用。仅是关于直播的服务。
抽取并剪裁接口	采用。用于统一不同的数据格式。

➤ 可修改性

可选的设计决策或模式	决策理由
划分模块	采用。是一种常用且有效的策略。
抽象通用服务	不采用。并没有通用服务需要被抽象。
添加接口	采用。增加可修改性常用的策略。
运行时注册	不采用。会带来额外的管理开销。

➤ 负载性能

可选的设计决策或模式	决策理由
维持计算的多个副本	不采用。考虑同步需要的代价太大。
增加可用资源	不采用。会增加较多成本。
区分校内校外网络接入	采用。适用于此类开发成本不应太大的系统。

➤ 实时性

可选的设计决策或模式	决策理由
使用 RTMP (Real Time Messaging Protocol) 协议	不采用。并没有很好的兼容性。
使用 HLS (HTTP Live Streaming) 协议	采用。该技术成熟且容易使用。此外，开源有益于控制开发系

Streaming) 协议	统的成本。
---------------	-------

9.2.4 第二次迭代结果

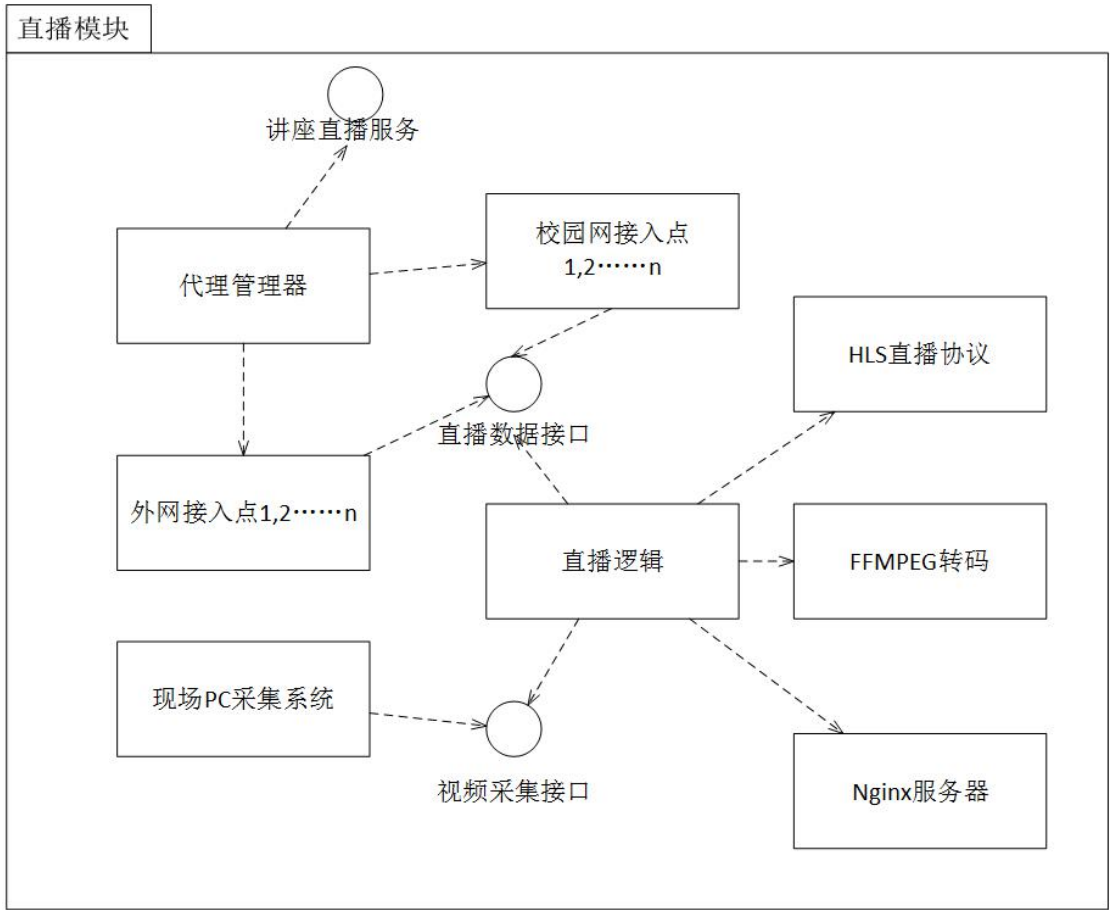


图 10：SOA 第二次迭代结果

9.3 第三次迭代过程

选取讲座报名模块作为系统元素进行分解。在质量属性方面，该模块与可获得性、可修改性、负载性能有密切关系。

9.3.1 识别所选模块的 ASR

#	Architectural Drivers	Importance	Difficulty
1	Scenario 2: 可获得性	high	medium
2	Scenario 4: 可修改性	medium	low
3	Scenario 5: 负载性能: 系统能够允许 500 个用户同时进行正	high	high

	常的访问、报名操作		
--	-----------	--	--

9.3.2 每个 ASR 可选的设计决策

➤ 可获得性

可选的设计决策或模式	分析
命令/响应	一个组件发出一个命令，并希望在预定义的时间内收到一个来自审查组件的响应。与对所有进程发出命令的远程错误探测器相比，这种策略所使用的通信带宽更少
心跳	一个组件定期发出一个心跳信息，另一个组件收听该信息。可以在起到心跳的同时传递数据。
主动冗余（热启动）	所有的冗余组件都以并行的方式对事件作出响应。错误发生时，使用该战术的系统的停机时间通常是几毫秒，因为备份是最新的，所以恢复所需的时间就是切换时间。
被动冗余	一个组件（主要的）对事件作出响应，并通知其他组件（备用的）必须进行的状态更新。该战术依赖于能够可靠地接管工作的备用组件。停机时间通常为几秒钟。
备件	备用件是计算平台配置用于更换各种不同的故障组件。当出现故障时，必须将其重新启动为适当的软件配置，并对其状态进行初始化。该战术的停机时间通常为几分钟。

➤ 可修改性

可选的设计决策或模式	分析
划分模块	可以有效缩小要修改的范围
抽象通用服务	将多个模块都需要的服务提取出来，形成可供多个模块共同使用的通用服务
添加接口	将功能和实现分离，隔离需要更改的模块
运行时注册	支持即插即用操作，但需要管理注册的额外开销

➤ 负载性能

可选的设计决策或模式	分析
维持计算的多个副本	需要考虑如何使副本保持一致和同步
增加可用资源	使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络。但是会增加成本

9.3.3 设计决策的选择及分析

➤ 可获得性

可选的设计决策或模式	决策理由
命令/响应	不采用。
心跳	采用。常用的方式，且使系统的可获得性更高。
主动冗余（热启动）	采用。该系统对于选课报名的可获得性要求很高，采用热备份可以获得更高的可获得性。
被动冗余	不采用。停机时间过长，对于讲座报名这种短时间内对可获得性要求极高的子系统而言并不够好。
备件	不采用。理由同上。

➤ 可修改性

可选的设计决策或模式	决策理由
划分模块	采用。是一种常用且有效的策略。
抽象通用服务	不采用。并没有通用服务需要被抽象。
添加接口	不采用。不适用于该子系统。
运行时注册	不采用。会带来额外的管理开销。

➤ 负载性能

可选的设计决策或模式	决策理由
维持计算的多个副本	采用。保证在一个计算逻辑失效后可以在较短的时间内启用备用逻辑。
增加可用资源	不采用。会增加较多成本。

9.3.4 第三次迭代结果

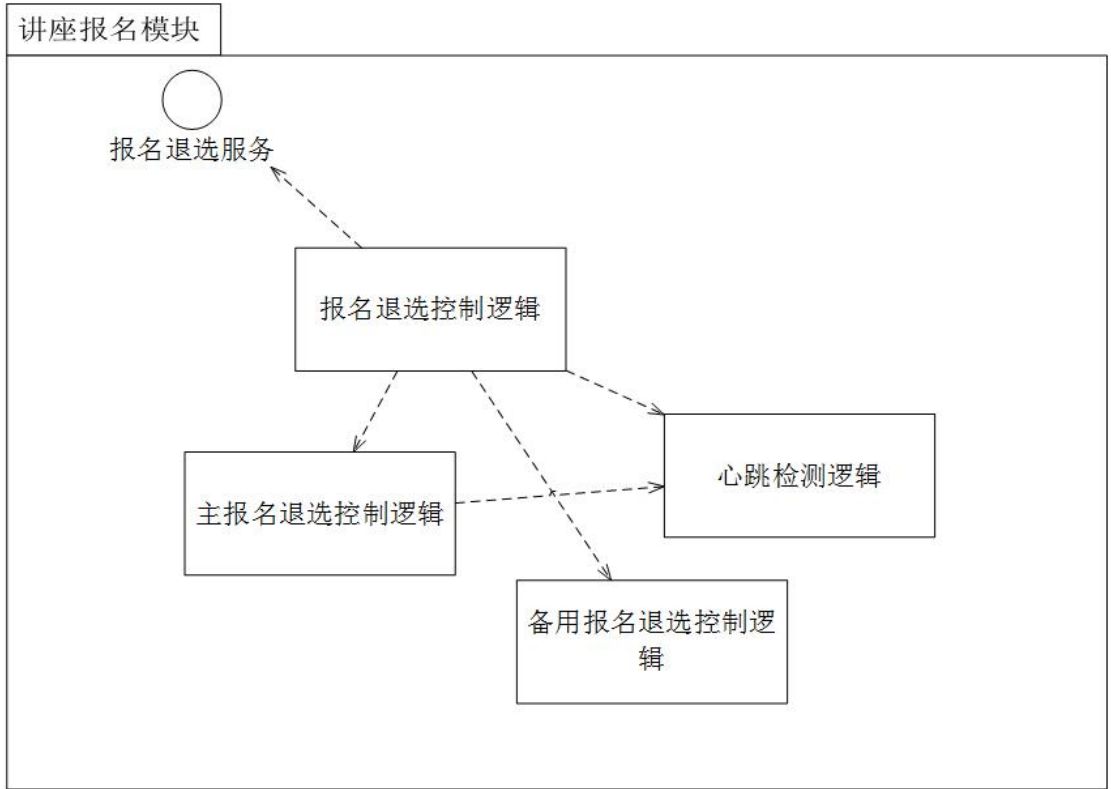


图 11：SOA 第三次迭代结果

9.4 第四次迭代之后

受篇幅所限，第四次迭代直至完成架构的过程此处不再给出，最终结果请参照第 2 节。

10.ATAM 分析过程

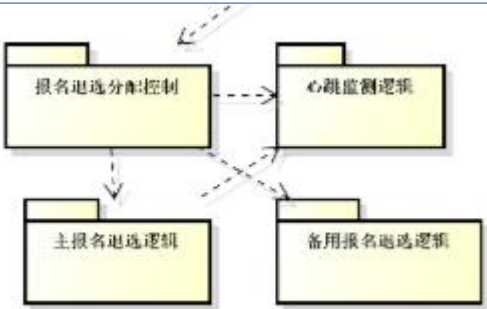
10.1 质量属性效用树

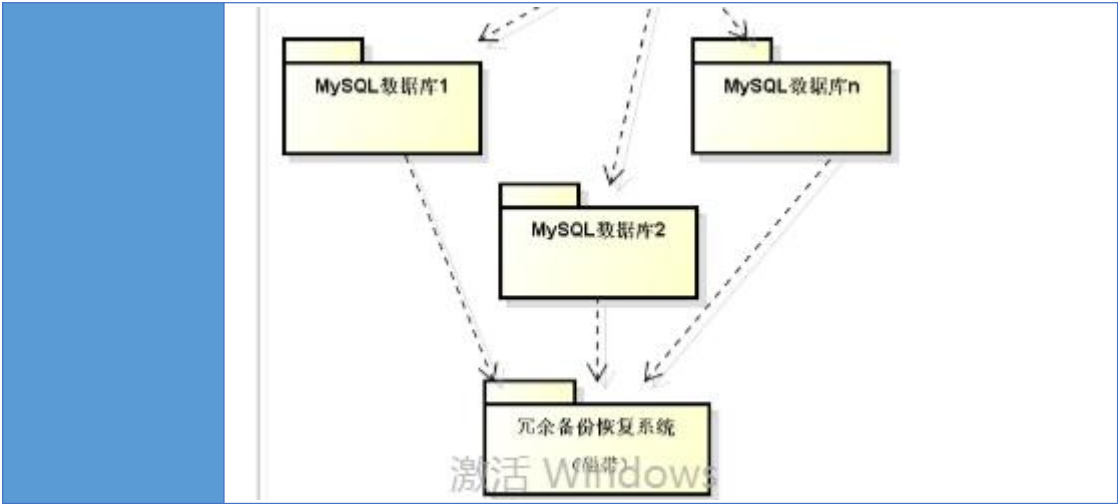
质量属性	属性求精	场景
安全性	数据加密	A1:用户数据被窃取，窃取者无法破译加密后的用户密码等敏感信息(H,H)
		A2:定期更换加密密钥，当窃取者破解之前的加密密钥

		时，密钥已更换，窃取者仍然无法破译加密后的用户密码等敏感信息(M,M)
	身份验证	A3:某用户未通过身份验证试图向探测器发送控制指令和访问图像数据，被系统拒绝访问(H,M)
	攻击侦测和防御	A4:系统受到恶意攻击，系统侦测到攻击后，记录攻击日志，提醒管理员并锁住数据，直到确认安全(M,L)
可用性	故障监测	A5:采用命令/响应，一个组件发出一个命令，并在 0.5s 内收到一个来自审查组件的响应(M,H)
		A6:采用心跳机制，系统某部分发生故障失去心跳后，系统在 5min 内检测到故障源(H,H)
	功能备份	A7:当检测到故障后，系统在 1s 内切换至备用控制逻辑，恢复正常功能(H,H)
	故障重启	A8:其他部分发生故障，若不能恢复，在 5min 内通过重启恢复正常功能(H,M)
互操作性		A9:当系统发起/收到与校园卡系统/教务网系统/现场 PC 端视频采集系统的请求/响应时，不需要经过数据格式转换，交换信息数据成功的概率不低于 99%(H,H)
可修改性		A10:当与系统交互的教务系统，校园卡系统提供的接口发生修改时，与之相关的模块修改在 1 人月内完成(M,M)
		A11:开发者更新某些系统模块的功能，修改预算未超过总预算的 10%，其他无关的系统功能未受影响(H,L)

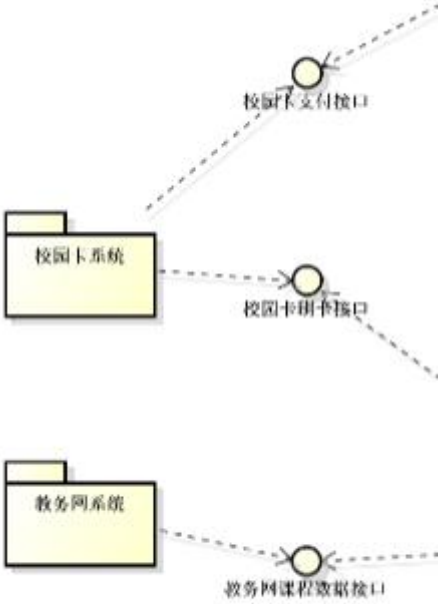
性能	负载	A12:系统至少允许 500 个用户同时进行正常的访问 (H,H)
		A13:报名操作/观看讲座直播用户访问所需页面的时间不超过 0.1s(H,M)
	容量	A14:系统至少能够存储 50T 的视频数据(H,L)
	实时性	A15:当有大量用户同时观看直播时，系统直播延时不超过 5s(H,H)
可维护性		A16:系统开发者希望修改加密密钥时，修改时间不超过 1 人日(M,L)

10.2ATAM 分析

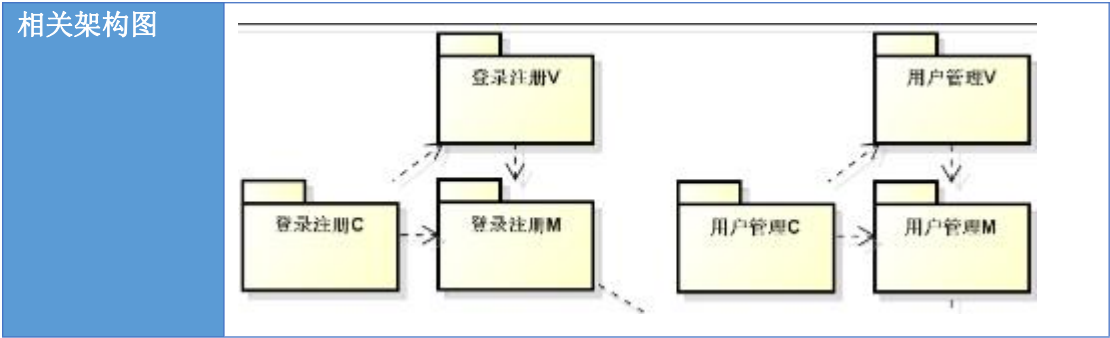
场景：A1	经过授权的个人用户希望进入系统报名参与讲座，但系统无法提供正确的反馈，系统记录错误日志并在 5s 之内从错误中恢复			
质量属性	可用性			
环境	联网状态			
刺激	希望进入系统报名参与讲座时系统无法提供正确的反馈			
响应	系统记录错误日志并从错误中恢复			
架构决策	敏感点	权衡点	风险	非风险
心跳	S4	T6	R2	
主动冗余	S5	T2	R3	
理由说明	系统出错后恢复的速度会极大影响用户体验，心跳可在 2s 内检测到故障，主动冗余可以保证在出现故障后，系统的停机时间只有几毫秒。虽然会降低部分性能，但提高了可用性，也是值得的。			
相关架构图				

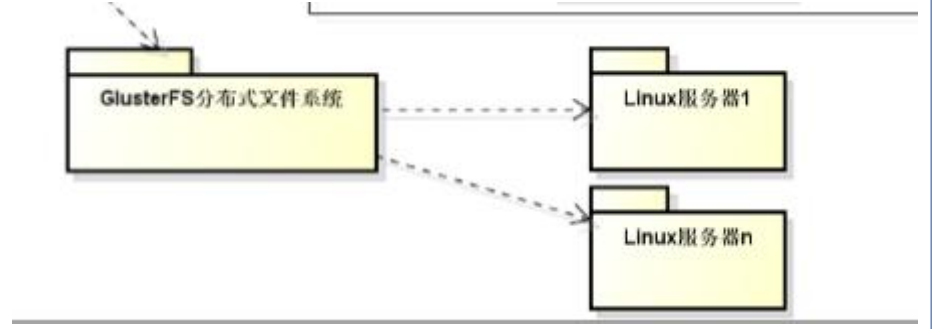


场景：A2	系统受到恶意攻击，系统侦测到攻击后，防火墙会识别攻击并阻断攻击			
质量属性	安全性			
环境	运行时			
刺激	外部攻击			
响应	系统侦测到攻击后，防火墙识别攻击并阻断攻击			
架构决策	敏感点	权衡点	风险	非风险
在系统服务收到访问时验证用户身份，授权给用户应有的权限	S2			N2
自动攻击侦测	S3	T1	R1	
使用 DDOS 分防火墙	S7			N3
理由说明	<p>本项目与学生的校园卡相关，并于教务网相关联，所以安全性尤为重要，所以对用户进行权限设置。</p> <p>攻击者若不能通过用户认证，可能使用各种方式对系统发动攻击。为安全起见，自动侦测攻击并使用防火墙阻止攻击。虽然可能降低部分性能，但是这种风险是可以接受的，而泄密的风险是不可接受的。</p>			
相关架构图	<p>The diagram shows a vertical flow of three boxes. The top box is '用户浏览器'. A dashed arrow labeled '访问' points down to the middle box, '黑利DDOS防火墙 (免费)'. Another dashed arrow labeled '访问' points down from the middle box to the bottom box, '所有View'.</p>			

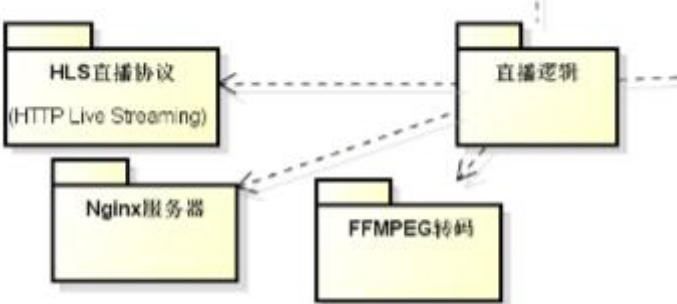
场景：A3	系统发起/收到与校园卡系统/教务网系统/现场PC端采集系统的请求/响应，希望与其进行请求/响应数据交换			
质量属性	互操作性			
环境	运行时			
刺激	系统希望与外界系统进行请求/响应数据交换			
响应	请求被成功获取并成功交换数据			
架构决策	敏感点	权衡点	风险	非风险
抽取并裁剪接口	S13			N6
理由说明	本项目拥有校园卡支付，校园卡签到，教务网信息导入等功能，所以互操作性是很重要的质量属性。采用合适的接口，可以方便与外界进行数据交换，提高互操作性。			
相关架构图				

场景：A4	开发者修改系统用户界面、数据标准、控制逻辑等			
质量属性	可修改性			
环境	设计、开发系统时			
刺激	开发者希望修改系统用户界面、数据标准、控制逻辑等			
响应	需要修改的模块被正确修改，并不影响其他功能的实现			
架构决策	敏感点	权衡点	风险	非风险
划分模块	S7			N4
独立存储密钥数据的模块	S11			N5
理由说明	对于一个复杂的系统来说，系统的可修改性和可维护性都是非常重要的。采用模块划分的决策，增强内聚，可以有效缩小要修改的范围，是成熟的设计原则，可增强可修改性。并独立存储密钥数据的模块，可提高可维护性			



场景：A5	有 500 名用户同时在系统上进行讲座报名操作/查看讲座直播			
质量属性	性能（负载）			
环境	联网状态			
刺激	学生用户希望在系统上进行讲座报名操作/查看讲座直播			
响应	系统能够正常工作，为每一个用户提供相应			
架构决策	敏感点	权衡点	风险	非风险
增加可用资源	S8		R4	
使用分布式数据库服务器	S9	T3	R5	
理由说明	使用速度更快的处理器、额外的处理资源、额外的内存、更快的网络，可以保证在负载方面提高系统性能。但这会使得成本偏高，造成风险。因此需要在可以负担的范围内尽可能增加可用资源。分布式数据服务器的性价比显著优于单台商业中型机和大型机，能够提供较为优质的数据服务，提高性能。			
相关架构图				

场景：A6	用户在系统上观看讲座直播			
质量属性	性能			
环境	运行时			
刺激	学生用户希望在系统上观看讲座直播			
响应	系统直播视频清晰流畅，延时短			
架构决策	敏感点	权衡点	风险	非风险
使用 HLS（HTTP Live Streaming）协议	S10	T4	R6	
使用 FFMPEG 转	S14			N7

码				
理由说明	使用 HLS(HTTP LIVE STREAMING)协议，延时虽然较高，但仍在可接受的范围内。且兼容性比较好，提高了可移植性。在性能和可移植性兼备的情况下，选择了 HLS 新协议。使用 FFMPEG 转码，可以保证视频质量，同时提高了可移植性			
相关架构图				

10.3 敏感点和权衡点

10.3.1 敏感点

S1	对用户信息进行加密	保护用户信息，影响了安全性(正面)，是 安全性的敏感点
S2	在系统服务收到访问时验证用户身份，授权给用户应有的权限	有利于保证系统服务仅受到合法访问，是 安全性的敏感点
S3	自动攻击侦测	出现攻击时可以自动侦测到，提高了安全性，是 安全性的敏感点 ，同时会增加服务器负担，降低性能，是 性能的敏感点
S4	心跳	一个组件定期发出一个心跳信息，另一个组件收听该信息。可以用来识别错误，是 可用性的敏感点 。同时可能占用资源，导致性能的下降，是 性能的敏感点
S5	主动冗余	所有的冗余组件都以并行的方式对事件作出响应，错误发生时，使用该战术的系统停机时间通常是几毫秒，是 可用性的敏感点 。但备份占用了系统资源，影响了性能，是 性能的敏感点
S6	使用 DDOS 分防火墙	对各种常见的攻击行为进行识别，并通过集成的机制实时对这些攻击流量进行处理及阻断，是 安全性的敏感点 。
S7	划分模块	将系统模块化，在修改时可以有效缩小要修改的范围，是 可修改性的敏感点
S8	增加可用资源	使用速度更快的处理器、额外的处理资源、额外的内存、速度更快的网络，可提高性能，是 性能的敏感点 。但会影响成本
S9	采用分布式数据服务器	分布式数据服务器的性价比显著优于单台商业中型机和大型机，能够提供较为优质的数据服务，是 性能的敏感点（正面） 。但是由于数据分散存储，技术门槛较高，对维护工作，特别是数据完整性、一致性的维护提出了较高的要求，是 可维护性的敏感点（负面）

S10	使用 HLS(HTTP Live Streaming) 协议	兼容性较好, 提高了可移植性, 是 可移植性的敏感点 。但延时比较高, 影响了性能, 是 性能的敏感点 。
S11	独立存储密钥数据的模块	将密钥数据独立存储, 可以提高可修改性, 是 可修改性的敏感点
S12	采用 GlusterFS 分布式文件系统	具有线性横向拓展能力, 提高了可拓展性, 是 可拓展性的敏感点 。增加了客户端的负载, 占用了相当的 CPU 和内存, 降低了性能, 是 性能的敏感点 。
S13	抽取并裁剪接口	针对请求响应交换的数据不一致情况, 分别通过接口与外部系统相交互, 是 互操作性的敏感点
S14	使用 FFMPEG 转码	FFMPEG 转码提供了录制、转换以及流化音视频的完整解决方案, 有助于播放清晰的视频, 是 性能的敏感点(正面) , 同时它包含了非常先进的音频/视频编解码库 libavcodec, 保证了高可移植性, 是 可移植性的敏感点

10.3.2 权衡点

T1	自动攻击侦测	提高了安全性, 但会增加服务器负担, 降低性能, 是 安全性和性能的权衡点
T2	主动冗余	错误发生后恢复时间短, 提高了可用性, 但备份占用了系统资源, 影响了性能, 是 可用性和性能的权衡点
T3	分布式数据服务器	性价比高, 以相同成本提供更佳的容量和读写速率, 但是维护工作技术门槛高, 是 性能和可维护性的权衡点
T4	使用 gHLS(HTTP Live Streaming)协议	兼容性较好, 提高了可移植性, 但延时比较高, 影响了性能, 是 性能和可移植性的权衡点
T5	采用 GlusterFS 分布式文件系统	提高了可拓展性, 同时降低了性能, 是 可拓展性和性能的权衡点
T6	心跳	提高了可用性, 但降低了性能, 是 可用性和性能的权衡点

10.4 风险和非风险

10.4.1 风险

R1	自动攻击侦测	攻击不经常发生, 系统一直自动侦测攻击, 可能导致对资源的很大浪费, 影响性能
R2	心跳	可能造成性能负担, 影响系统性能
R3	主动冗余	备份占用了系统资源, 可能会影响系统性能
R4	增加可用资源	可提高性能, 但成本较高
R5	分布式数据服务器	分布式数据服务器技术门槛高, 提高人力成本。如果维护不当, 数据的完整性和一致性可能出现问题的。
R6	使用 gHLS(HTTP Live Streaming)协议	延时比较高, 可能影响系统性能
R7	采用 GlusterFS 分布式文件系统	可能会增加客户端的负载, 占用相当的 CPU 和内存, 降低性能

10.4.2 非风险

N1	对用户信息进行加密	多用户系统和涉密系统的标准配置
N2	在系统服务收到访问时验证用户身份，授权给用户应有的权限	多用户系统和涉密系统的标准配置
N3	使用 DDOS 分防火墙	可以有效识别并阻断攻击，有效提高安全性
N4	划分模块	成熟的设计原则，显著利于增强可修改性、可移植性
N5	独立存储密钥数据的模块	同上
N6	抽取并裁剪接口	通过接口与外部系统交互，提高了互操作性
N7	使用 FFMPEG 转码	提高了性能和可移植性

11.挑战和经验

在本次项目实践中，针对校园讲座管理系统所需的诸多功能性需求和非功能性需求，我们将安全性、可获得性、互操作性、可修改性、性能和可维护性列为本次架构设计需要着重考虑的方面，提出了 MVC 以及 SOA 两种候选方案，由于系统对安全性以及成本有所限制，最终选择了 MVC。

我们认为在这次实践中遇到的挑战主要有以下几点：首先，对于一些前人没有做过的新系统，通常难以一下子设计出合适的架构。在架构初期，通常都要经历一个不断探索的阶段；其次，在对 ASR 提出设计决策时，不可避免的会涉及到一些不熟悉的领域（比如此次的视频直播），这就需要我们快速了解一个陌生领域，从而提出合适的解决方案。

通过这次实践，我们掌握了一个系统架构的完整的设计流程，对架构设计方法 ADD 和架构评估方法 ATAM 有了更加深刻的理解：ADD 是一个递归的分解过程，在每个阶段都选择战术和架构模式来满足一组质量属性场景，然后对功能进行分配，以实例化由该模式所提供的模块类型；ATAM 作为一种架构评估的综合方法，需要我们根据质量属性需求列表以及每个高优先级场景相关的构架决策，判断这些决策是否有风险，从而找到架构中任何存在问题的地方，有利于构建一个健壮的软件架构。

以下为团队负责人的个人感想：经过实践，我充分认识到架构活动深深触及了软件系统“本质上的困难”即复杂度，软件系统作为一个有机整体，其架构设计和分析活动具有本质意义上的困难性和挑战性。也是由于这种原因，架构活动的各个阶段难以分工和并行进行。认识到这种困难，并激发出对这种挑战性工作的兴趣，是我从课程实践中得到的最大的收获。

12. 组员和分工

组员	学号	分工
梁思宇	131250129	在本次作业中负责对软件架构进行评估，采用了 ATAM 方法，分析构架方法，列出优先级场景，把相关构架决策编成文档，确定其有风险决策和无风险决策，敏感点和权衡点，并对其进行分类，以确信该方法的实例化适合满足所要达到的质量属性需求，最终进行归档。
邹卓晋	131250158	在本次作业中，我主要负责原型中 MVC 框架的搭建以及后端 model 和 controller 部分的编写。在代码编写完成后，我进行了原型的演示以及用户使用说明的编写。
曾婧	131250159	在这次项目中，经过商定决定选择了 MVC 的架构方法后，我负责了编写 View 层的部分代码，以及 View 层与 Controller 层的数据交互的代码，同时原型的页面设计。与需求撰写人员共同讨论了需求是否合理，如何实现等问题。同时还与负责 Controller 层的同学共同制定了接口，包括接口的命名、规范和详细参数。
吴超月	131250168	<ol style="list-style-type: none"> 1、负责本文中功能点操作场景的文档编写 2、负责非功能性需求和 ASR 的场景描述及文档编写 3、负责 MVC 和 SOA 两种架构的模块视图绘制 4、负责两个架构的 ADD 过程中每个 ASR 可选取的设计决策的编写 5、负责 MVC 架构的 ADD 过程及文档编写，具体包括三个迭代，在每次迭代过程中对所选模块进行 ASR 的识别，进行设计决策的选择及分析，并进行每次迭代结果的产物图绘制 6、负责本文档内容的整合和样式的统一、整理
罗瑶	131250177	在本次大作业中，我主要负责我们系统的 ADD 过程，部分参与 ATAM 过程。首先我开始对系统采用 SOA 架构进行 ADD 过程，一共进行 3 个迭代：分别对直播模块和补退选模块进行识别所选模块的 ASR，进行设计决策的选择及分析；之后在 ATAM 过程中，我根据之前的开发画出了系统效用树；最后我进行了 MVC 和 SOA 两个架构选择的比较，最后选择出 MVC 架构。

陈云龙	131250181	在本次作业中，我主要负责原型设计、Controllor 层、Model 层和部分 View 层代码的编写、调试、项目部署与维护，根据 MVC 架构和 SOA 架构的比较结果选择 MVC 架构设计代码，以及系统 UML 的类图设计和绘图、组件/连接器与实现类的 mapping，参与系统需求、功能，架构的讨论，和其他原型设计人员定义接口与前后端的交互。
倪小凡	131250185	在这次项目中，我负责了编写 view 层的部分代码。此外，还根据项目架构设计过程完成了整个项目设计过程中关于挑战和经验的总结。
丁霄汉	131250207	<p>在团队项目中，我担任团队负责人，所做的工作主要包括分工，组织会议和讨论，架构模式的分析、选择和绘图，具体实现策略（如 Gluster 文件系统、HLS 协议）的分析、选择，课堂 PPT 演示，以及对应的文档工作。</p> <p>在提交产物中，我所贡献的部分包括：proposal 和最终报告中的 C&C 视图，课堂演示所用的 PPT，最终报告中的“两种架构模式的比较及最终选择”以及“具体实现技术的选择与解释”相关部分。</p>